# Improving Reinforcement Learning Agents Using Genetic Algorithms

Akram Beigi, Hamid Parvin, Nasser Mozayani, and Behrouz Minaei

Computer Engineering School, Iran University of Science and Technology (IUST), Tehran, Iran
{Beigi,Parvin,Mozayani,B_Minaei}@iust.ac.ir

**Abstract.** In this paper a new Reinforcement Learning algorithm was proposed. Q learning is a useful algorithm for agent learning in nondeterministic environment but it is a time consuming algorithm. The presented work applies an evolutionary algorithm for improving Reinforcement Learning algorithm.

## 1 Introduction

Reinforcement learning is a computational approach to building agents that learn their behaviors by interacting with an environment (Sutton and Barto, 1998). A *reinforcement learning agent* senses and acts in its environment in order to learn to choose optimal actions to achieve its goal. It has to discover by trial-and-error search how to act in a given environment. For example, a robot may have sensors to perceive the environment state, and actions to change its state such as moving in different directions. For each action the agent receives feedback (also referred to as a reward or reinforcement) to distinguish what is good and what is bad. The agent's task is to learn a policy or control strategy for choosing the best actions in such a long run that achieve its goal. For such a purpose the agent stores a *cumulative reward* for each state or state-action pair. The ultimate objective of a learning agent is to maximize the cumulative reward it receives in the long run, from the current state and all subsequent next states along with goal state.

Reinforcement learning systems have four main elements (Cuay´ahuitl, 2009): a policy, a reward function, a value function, and optionally, a model of the environment. A *policy* defines the behavior of the learning agent. It consists of a mapping from states to actions. A *reward function* specifies how good the chosen actions are. It maps each perceived state-action pair to a single numerical reward. A *value function* specifies what is good in the long run. The value of a given state is the total reward accumulated in the future, starting from that state. Approaches based on value function attempt to find a policy that maximize the return by maintaining a set of estimates of expected returns for one policy. Using of value functions distinguish reinforcement learning methods from evolutionary methods that search directly in policy space guided by scalar evaluations of entire policies. The *model of the environment* is something that mimics the environment's behavior. A simulated model of the environment may predict the next environment state from the current state and action. The environment is

```
Q-Learning Algorithm:
Initialize $Q(s,a)$ arbitrarily
Repeat (for each episode):
Initialize $s$
Repeat (for each step of episode):
        Choose $a$ from $s$ using policy derived from $Q$
                (e.g., ε-greedy)
        Take action $a$, observe $r$, $s'$
        $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
$s \leftarrow s'$
Until $s$ is terminal
```

**Fig. 1.** Q- Learning Algorithm

usually represented as a Markov Decision Process (MDP) or as a Partially Observable MDP (POMDP) (Vidal 2007; Sutton and Barto, 1998; Shoham et al. 2009).

One of the most important reinforcement learning algorithms is Q-learning. In this case, the learned action-value function, Q, directly approximates Q*, the optimal action-value function, independent of the policy being followed. This simplifies the analysis of the algorithm and enabled early convergence proofs. The Q-learning algorithm is shown in procedural form in Figure 1.

### 1.1 Markov Decision Process

We have assumed that the agent senses the state of the world then takes an action which leads it to a new state. We also make the further simplifying assumption that the choice of the new state therefore depends only on the agent's current state and the agent's action. This idea is formally captured by a Markov decision process or MDP.

An MDP is defined as a 4-tuple <*S*,*A*,*T*,*R*> characterized as follows: S is a set of states in the environment, A is the set of actions available in the environment, T is a state transition function in state *s* and action *a*, R is the reward function.

The optimal solution for an MDP is that of taking the best action available in a state, i.e. the action that collected as much reward as possible over time.

### 1.2 Multi-task Learning

*Multi-task learning* (Wilson et al., 2007) is an approach to machine learning that learns a problem together with other related problems at the same time, using a shared representation. This often leads to a better model for the main task, because it allows the learner to use the commonality among the tasks.

Tanaka et al. (2003) proposed Multitask Reinforcement Learning Problems on a number of instances of the Markov Decision Processes, which are sampled from the same probability distributions. The instances of the Markov Decision Processes are assumed to be sequentially presented to learning agents so that the learning agents can effectively utilize knowledge acquired from past instances of the Markov Decision Processes.

If the size of problem space of such instances is huge, Evolutionary Computation could be more effective than Reinforcement Learning Algorithms and *memory-based Evolutionary Computation* (Handa, H. 2007) is used for storing past optimal solutions. Dynamic or uncertain environments are crucial issues for Evolutionary Computation since. They are expected to be effective approach in such environments.

The specific families of Reinforcement learning techniques we look at are derived from the Q-learning algorithm for learning in unknown MDPs.

Because of slowness in learning by Reinforcement learning, in this work we propose a modified Q-learning algorithm with applying memory-based Evolutionary Computation technique for improving learning in multi task agents.

## 2 Memory-Based Evolutionary Computation

Handa, has used memory-based Evolutionary Computation for storing past optimal solutions. In that work, each individual in population denotes a policy for given task. At environmental change, the best individual in current population is interested into the archive. Then individuals in the archive are randomly selected and they are moved into the population. An overview of Memory-based Evolutionary Programming is depicted in Figure 2.

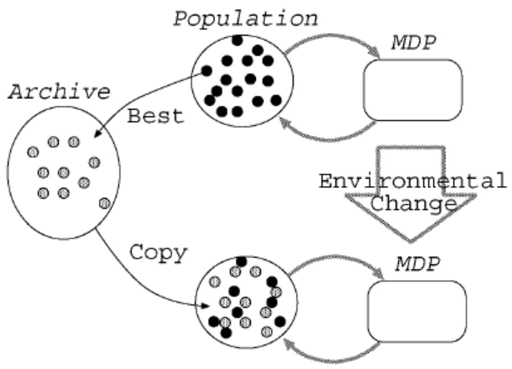

**Fig. 2.** An overview of Memory-based Evolutionary Programming

A large number of studies concerning dynamic/uncertain environments have been made; have used Evolutionary Computation algorithms (Goh, and Tan, 2009). The aim of these problems is to gain goal as fast as it could be. The significant issue is that the robots could get assistance from their previous experiences. In this paper a memory-based Evolutionary Computation for Multitask Reinforcement Learning problems is examined.

## 3 GA-Based RL (GARL) Algorithm

GARL (Jiang, J. 2007) is an approach for searching the optimal control policy for an RL problem by using GAs. When GAs are used for RL problems, the potential solutions are the policies and are expressed as chromosomes, which can be modified by genetic operations such as mutation and crossover.

GAs can directly learn decision policies without studying the model and state space of the environment in advance. The only feedback for GAs is the fitness values of different potential policies. In many cases, fitness function can be expressed as the sum of instant rewards, which are used to update the Q-values of value function-based RL algorithms.

```
GA-based Reinforcement Learning (GARL):
   Initialize: Construct a population pool randomly with a given policy form.
               The size of population is Nc (even number);
   Repeat until the terminal conditions are satisfied
      (1) Couple: Couple the Nc chromosomes randomly into Nc/2 groups.
                  The two chromosomes in each group are called parents;
      (2) For each group
         (a) Form a family:
             Produce children from parents using crossover() and mutation().
             The number of children is Nchildren.
             Parents and children form a family;
         (b) Evaluate:
             Calculate the fitness value of each member of the family by evaluation();
         (c) Select:
             Select two chromosomes in the family with the highest fitness values;
         (d) Replace:
             Replace the old parents of the family in the population pool with the two
             chromosomes selected;
         End each group;
      (3) Update all the chromosomes in the population pool,
      (4) Start a new generation with the evolutive chromosomes.
  End.
```

**Fig. 3.** GA-based Reinforcement Learning (*GARL*) Algorithm

## 4 Evolutionary Q-Learning

With applying GARL for reinforcement learning agent in nondeterministic environments, we propose a Q-learning method called Evolutionary Q-learning. The algorithm is presented in Figure 4.

The proposed algorithm is examined in nondeterministic maze.

```
Evolutionary Q Learning (EQL):
Initialize $Q(s,a)$ by zero
Repeat (for each generation):
        Repeat (for each episode):
                Initialize s
                Repeat (for each step of episode):
                        Choose a from s using policy derived from $Q$
                                (e.g., $\varepsilon$-greedy)
                        Take action a, observe r, s'
                        $s \leftarrow s'$
                Until s is terminal
                Add visited path as a Chromosome to Population
        Until population is complete
        Do Crossover() by CRate
        Evaluate the created Childs
        Do tournament Selection()
        Select the best individual for updating Q-Table as follows:
                $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
        Copy the best individual in next population
Until satisfying convergence
```

**Fig. 4.** One Evolutionary Q Learning (*EQL*) Algorithm

## 5 Problem Definition and Modeling

Assume that a number of robots are working in a mine and their task is to search for gold from an initial point. The mine has a group of corridors which robots can pass through them. In specific paths there exist some barriers which do not let robots to continue. Now, suppose that because of decadent corridors, it is possible that in some places, there could be some pits.

If a robot enters to such pits, it could be don't able exit from that pit with possibility above zero by some moves and if it fails to exit by those moves, it should try again.

The aim of robots is finding the gold state as soon as possible. Note that the robots can use their past experiences.

### 5.1 Problem Space

In this work, an extension of Sutton's maze problem is used. As depicted in Figure below, the Sutton's Maze problem consist of 6 x 9 cells. This problem is composed of 46 states, which exclude 1 goal state and 7 collision cells (gray cells). Agents can take four kinds of actions; Up, Down, Left, and Right. The original Sutton's Maze problem is a deterministic problem (Sutton and Barto, 1998).

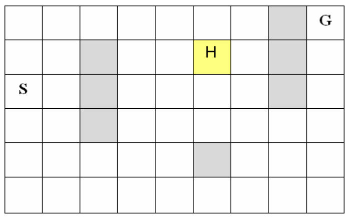

**Fig. 5.** Modified Sutton's maze

In nondeterministic version of this problem there are several probabilistic cells (cells with H letter). These probabilistic cells indicate that agents cannot move in accordance with their actions with a certain probability. That is, they have to stay the same cell with above zero probability; this probability is sampled from Normal distribution with average = 0, and variance = 1.

Agents will receive a reward+1, if they have get goal state. The sizes of population and archive are set to 100.

## 5.2 Actions

Each action of agent could be an MDP sample such as delineated in figure 6.

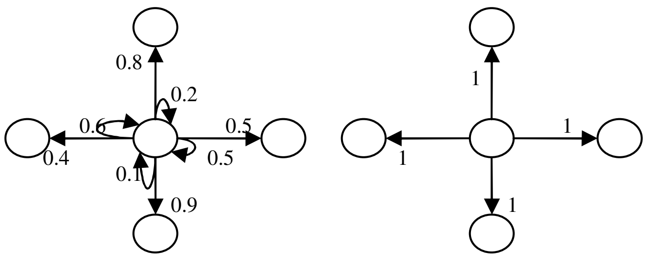

**Fig. 6.** Two sample actions are presented by MDP models

Right part shows certain case which in it agents move to next state by choosing any possible action with probability equal 1. On the other hand left part reveals that it is possible that the agents can not move to next state by choosing any possible action and it would remain in its position. for some do actions.

These MDPs presented to learning algorithm sequentially. The presentation time of each problem instance is enough to learn. The problem is to maximize the total acquired rewards for lifespan. Agents return to start state after arriving goal state.

## 5.3 Learning Algorithm

Considering the type of problems mentioned above and the fact that the reward is given to the agent only if it reaches to the goal, using original Q-learning algorithm may result in slowing down the learning process and the convergence might be decelerated. In addition in such problems, the shortest path may be not the best path; because of it may have some pit cells and it leads agent acts many movement until finding goal state. Therefore the optimum path has less pit cells and short length.

So applying an evolutionary version of Q-learning is more useful. Applied Items in this algorithm is as follow:

- **Chromosome's Structure:**
  Each chromosome is a array of states and actions, which agent visited them from start to goal state.
- **Value:**
  Value of each chromosome is the average of path length.
- **fitness:**
  There is proportion in fitness and inverse of value.
- **Crossover Operation:**
  A two point crossover is used. The figure 7 shows it.

**Crossover Algorithm:**

```
1. Select two individuals randomly
2. Select two states in first individuals randomly.
3. Seek first state from start state in second individual
4. Seek second state from last state in second individual
5. Swap these parts between individuals.
6. Evaluate these children
7. Keep the best.
```

**Fig. 7.** Crossover Algorithm

- **Selection Operation:**
  Truncation selection is applied.
- **Q-Table Updating:**
  After running generations, the best chromosome is selected for updating Q-Table as respect consisting states and actions.

## 6 Experimental Result

A single experiment is composed of 100 tasks, where each task consists of 100 generations. In this work 50 experiments have been done. Hence, 500,000 generations in 50 experiments have been executed. Table summarizes experiment results:

**Table 1.** Experimental Result

| | Best average of path length in population | worst average of path length in population | Total Average of path length |
|---|---|---|---|
| Original *QL* | 54.26 | 1600 | 449.1149 |
| Proposed *EQL* | 28.34 | 66.79 | 42.4738 |
| improvement | 47.7% | 95.8% | 90.5% |

## 7 Conclusion

Reinforcement Learning is good approach to building agents that learn their behaviors by interacting with an environment. Q-learning is a most important method in Reinforcement learning that it learns action-value function, directly approximates, the optimal action-value function, independent of the policy being followed.

Because of slowness in Q-learning, we applied genetic algorithms and used Memory based evolutionary computation for improving efficiency in nondeterministic environment. Proposed evolutionary Q-learning has about 90% improvement compared with original Q-learning algorithm in average case.

## References

1. Cuayáhuitl, H.: Hierarchical Reinforcement Learning for Spoken Dialogue Systems. PhD thesis, University of Edinburgh (2009)
2. Goh, K., Tan, K.: Evolutionary Multi-objective Optimization in Uncertain Environments. Springer, Heidelberg (2009)
3. Handa, H.: Evolutionary Computation on Multitask Reinforcement Learning Problems. In: 2007 IEEE International Conference on Networking, Sensing and Control, pp. 685–688 (2007)
4. Jiang, J.: A Framework for Aggregation of Multiple Reinforcement Learning Algorithms. PhD thesis, University of Waterloo (2007)
5. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments - a survey. IEEE Transaction Evolutionary Computation 9(3) (2005)
6. Shoham, Y., Leyton-Brown, K.: MULTIAGENT SYSTEMS Algorithmic, Game Theoretic, and Logical Foundations. Cambridge University Press, Cambridge (2009)
7. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
8. Tanaka, F., Yamaura, M.: Multi Task Reinforcement learning on the distribution of MDPs. In: Proceedings. 2003 IEEE International Symposium on Computational Intelligence In Robotics and Automation, September 2003, vol. 3, pp. 1108–1113. IEEE Press, Los Alamitos (September 2003)
9. Vidal, J.M.: Fundamentals of Multi Agent Systems (2009), `http://multiagent.com/2008/12/fundamentals-of-multiagent-systems.html`
10. Wilson, A., Fern, A., Ray, S., Tadepalli, P.: Multi-task reinforcement learning: A hierarchical Bayesian approach. In: International Conference on Machine Learning (ICML), Corvallis, OR, USA, pp. 1015–1022 (2007)